# Архитектура отказоустойчивого платежного шлюза

Павлов Алексей.
Wildberries






# Темы

- Чем не устраивал старый платежный шлюз
- Требования и ограничения при разработке нового
- Что в итоге по архитектуре/технологиям
- Протокол передачи данных в URL
- Дополнительные техники повышения SLA

# Ключевые цифры

- Больше 50 миллионов пользователей
- 8 миллионов посетителей ежедневно
- 14 стран (среди них Европа, США, Израиль)
- Около 1 миллиона транзакций в день
- Рост в 2 раза Y2Y

# Проблемы старого шлюза

- Отказоустойчивость (только в 1 дата-центр)
- Невозможность делать быстрые и частые обновления
- Сложность внедрения новых фич и интеграций
- Устаревший (для компании) стек
- Проблемы масштабирования

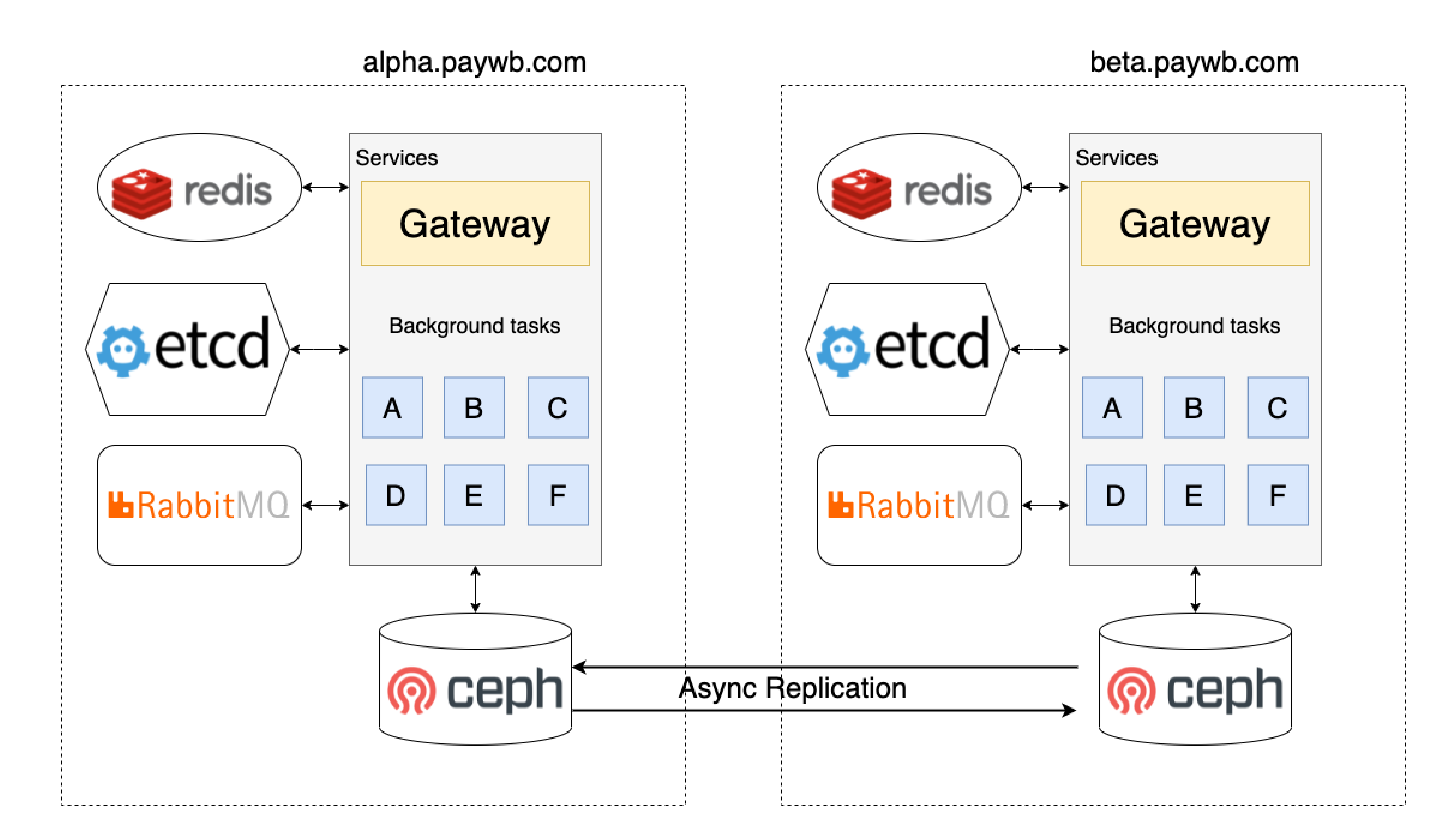
# Требования к новому шлюзу

- Толерантность к падению баз, дата-центров, банков и т.д.
- Способность принимать 100% оплат
- Выдерживать быстрый рост
- Быстрое внедрение новых фичей
- Мультитенантность

# Главные ограничения

- PCI DSS
- Быстрый выход в прод
- 2 дата-центра

# Архитектура



# CEPH

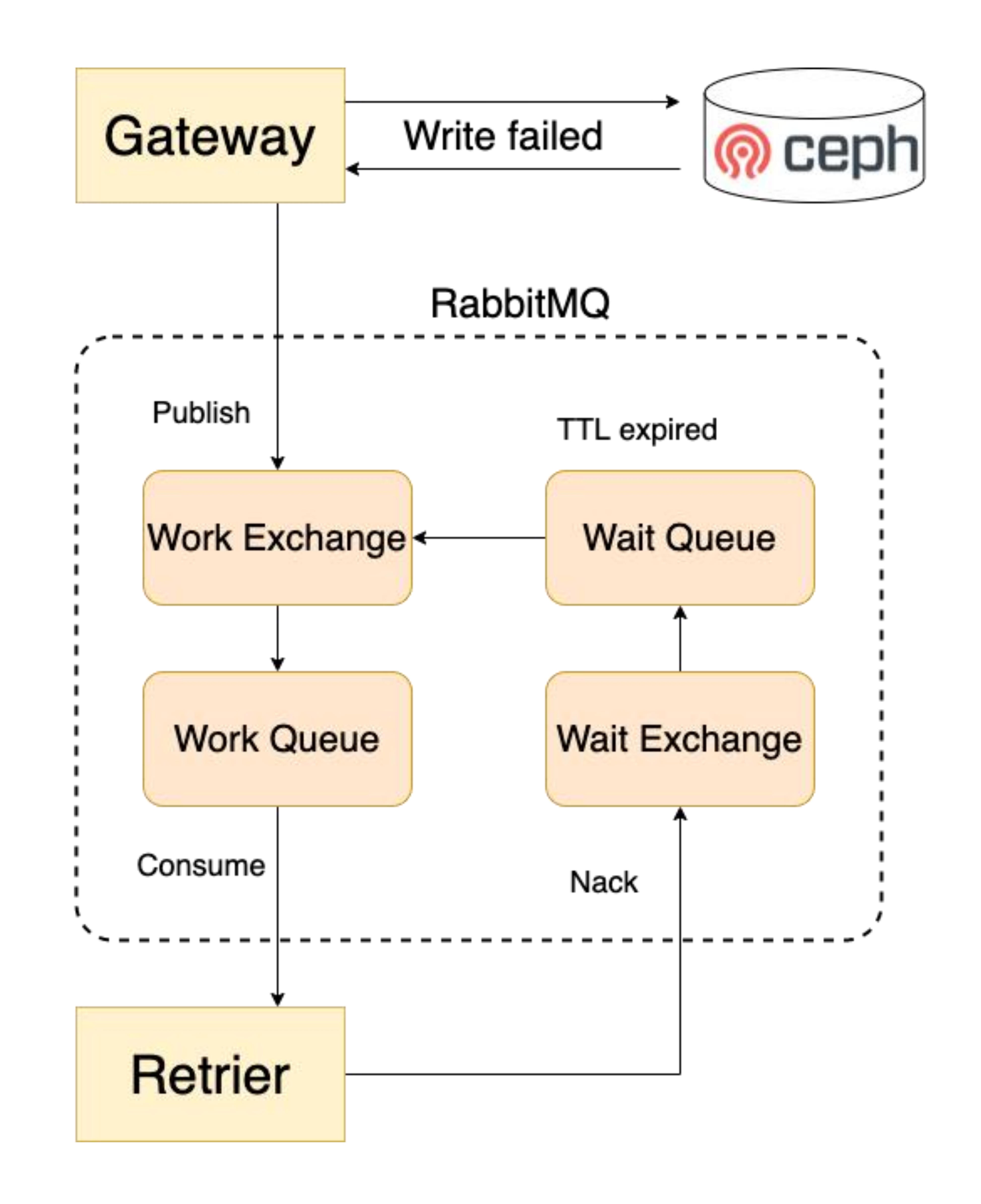
- Легко масштабироваться
- Можно не думать о данных
- S3-interface
- Экспертиза в компании
- Готовый инструмент для master-master-репликации

# RabbitMQ

1. Кладем задачу в очередь после неудачи какой-то операции

2. Крутим задачу в очередях до успеха или истечения определенного тайм-аута

3. При неудаче действия – кладем в очередь с большим TTL жизни (retry backoff)



# ETCD

- Защита от двойных списаний
- Rate-limiter
- Распределенный Mutex
- Дополнительная конфигурация сервисов

# Другие компоненты

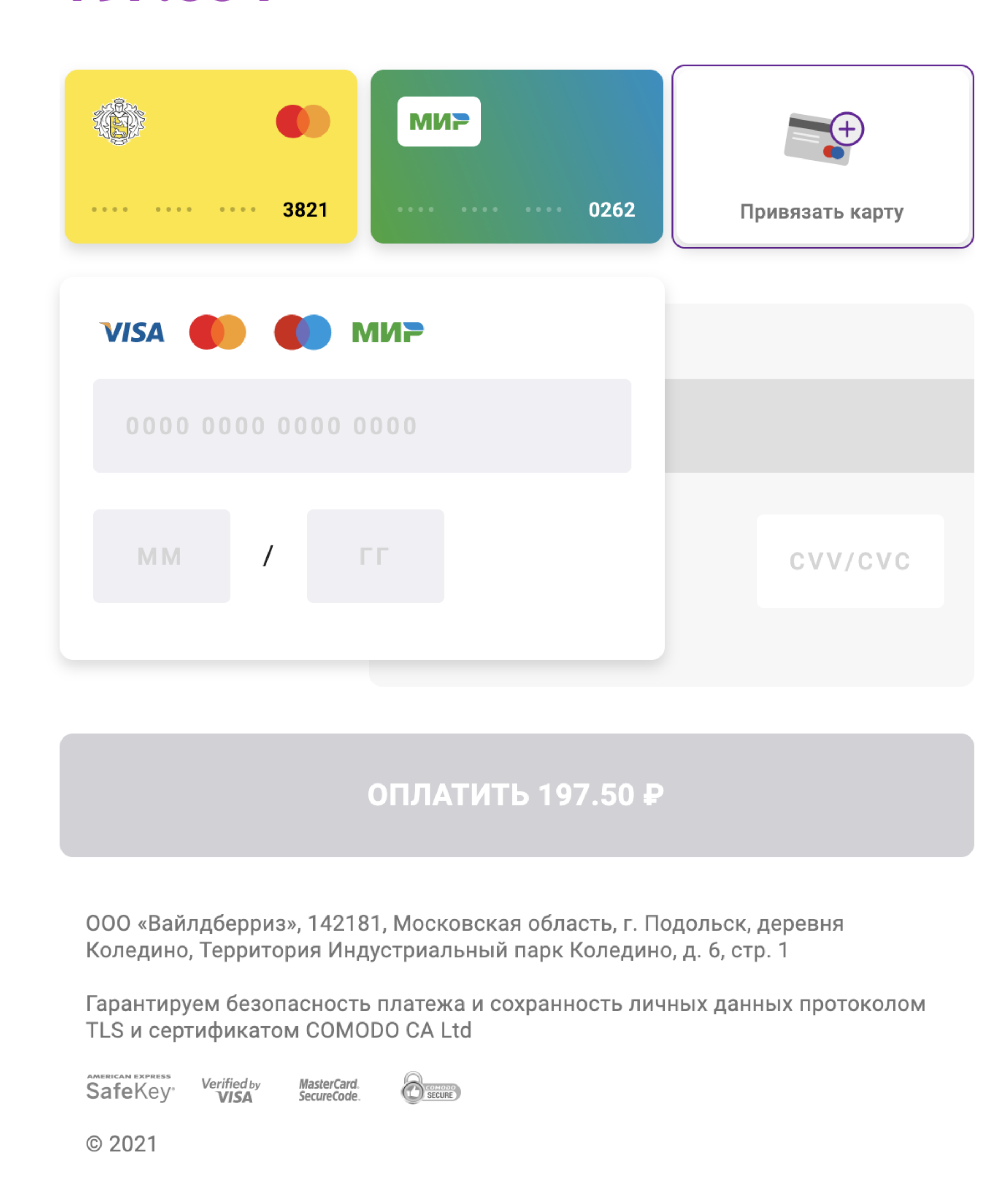
- Redis для кэширования запросов
- Vault, чтобы хранить секретные ключи

WILDBERRIES

# Оплата банковской картой

Итого с доставкой и НДС:

**197.50 ₽**



Выбор карты/ввод новой

# Хранение информации о транзакции

- Хранить только в базе – FAIL (зависимость от базы)
- Хранить еще где-то – FAIL (могут отказывать сразу несколько систем)
- Локальное хранилище – FAIL (мы не всегда контролируем клиента)
- Cookie – FAIL (cross-domain запросы)
- GET параметры – FAIL (могут быть обрезаны сторонними сервисами)
- Кодировать все в URL Path – SUCCESS

# Пример URL



Вызываемый метод
Пример необязательных параметров
Подпись параметров запроса
https://paywb.com/pl/request_payment/merchant/1/user/123456/use_3ds/true/.../expires_at/1620520690/signature/a2349cf2...
Опциональная локаль
Пример обязательных параметров
Время жизни ссылки в UNIX Timestamp

# Минусы протокола

- Поисковые движки не будут ранжировать
- Ограничения старых браузеров по длине
- Ограничение сторонних сервисов
- Нетипичный протокол, который требует своей реализации

# Библиотека для Go

```go
// URL-params model for rendering `request_payment` method page.
type Payment struct {
	/* Required */
	MerchantID   uint64         `urlpath:"merchant;required"`
	UserID       utils.UserID   `urlpath:"user;required"`
	Sum          uint64         `urlpath:"sum;required"`
	CurrencyCode uint64         `urlpath:"currency;required"`
	Transaction  string         `urlpath:"transaction;required"`
	ExpiresAt    int64          `urlpath:"expires_at;required"`
	ReturnURL    urlpath.Base64 `urlpath:"return_url"`
```

```go
func Example(path string) (string, error) {
	var request Payment
	err := urlpath.Unmarshal(path, &request)
	if err != nil {
		return "", err
	}

	params, err := urlpath.Marshal(request)
	if err != nil {
		return "", err
	}
	return request.formLink("/proceed_card" + params), nil
}
```



# Правила выбора банка

```
"Child": [
  {
    "Type": "mcc_tariffs",
    "RawRule": {
      "EnabledAcquirers": ["sber", "rsb", "tinkoff"],
      "ExtraRules": [
        {"Emitent": "homecredit", "Acquirer": "sber", "Enabled": true},
        {"PaymentSystem": "mir", "Emitent": "sber", "Acquirer": "sber", "Enabled": true},
        {"PaymentSystem": "mir", "Share": 500, "Is3DS": false, "Acquirer": "tinkoff", "Enabled": true},
        {"PaymentSystem": "mir", "Share": 500, "Acquirer": "rsb", "Enabled": true},
        {"PaymentSystem": "mir", "Acquirer": "sber", "Enabled": true},
        {"PaymentSystem": "mastercard", "Emitent": "tinkoff", "Is3DS": false, "Acquirer": "tinkoff", "Enabled": true},
        {"PaymentSystem": "mastercard", "Acquirer": "sber", "Enabled": true},
        {"PaymentSystem": "visa", "Emitent": "tinkoff", "Is3DS": false, "Acquirer": "tinkoff", "Enabled": true},
        {"PaymentSystem": "visa", "Emitent": "sber", "Acquirer": "sber", "Enabled": true},
        {"PaymentSystem": "visa", "Acquirer": "rsb", "Enabled": true}
      ]
    },
```

```
{
  "Type": "market",
  "RawRule": null,
  "Child": [
    {
      "Type": "uid_mod_range",
      "Rule": null,
      "RawRule": {
        "Ranges": {
          "{\"From\":0,\"To\":250}": 0,
          "{\"From\":250,\"To\":500}": 1,
          "{\"From\":500,\"To\":750}": 2,
          "{\"From\":750,\"To\":1000}": 3
        }
      },
      "Child": [
        {
          "Type": "sbermarket",
          "Fallback": "rsbmarket"
        },
        {
          "Type": "vbrrmarket",
          "Fallback": "sbermarket"
        },
        {
          "Type": "rsbmarket",
          "Fallback": "vbrrmarket"
        },
        {
          "Type": "tinkoffmarket",
          "Fallback": "sbermarket"
        }
      ]
    },
    {
      "Type": "rsb",
      "Fallback": "vbrr"
    }
  ]
},
```

# Circuit Breaker

- Переключения трафика между банками
- Выключение фичей и переход на стандартный флоу
- Отключение компонентов и форсирование резервных механизмов

# Резервные механизмы

- Брать информацию о транзакции из банка в случае отказа хранилища
- Дефолтная локальная конфигурация в случае недоступности динамической
- Идти на риск двойных оплат в случае невозможности проверить уникальность запроса
- Форсирование оплат новой картой в случае недоступности хранилища карт
- Локальная очередь запросов для добавления задач в RabbitMQ
- Дублирование финализации оплат при запросе ее статуса

# Организационные приемы

- Тестирование фич на отказ
- Учения с симулированием происшествий
- Разбор всех инцидентов
- База знаний «Что делать, если произошел X»

Вопросы?